CONTROL SYSTEM FOR PROGRAM EXECUTION

Patent Number: JP1010328
Publication date: 1989-01-13
Inventor(s): TAMARU HISASHI; others: 02
Applicant(s):: HITACHI LTD; others: 02
Requested Patent: ⌐ JP1010328
Application Number: JP19870165368 19870703
Priority Number(s):
IPC Classification: G06F9/06
EC Classification:
Equivalents:

Abstract

PURPOSE:To decide the propriety for execution of a low-order program in a process where a high-order program is turned into a pattern, by using a flag which displays a sorted/standardized working environment.
CONSTITUTION:A high-order program 1 searches group of entry table set in response to each low-order program 4 when the function contents are supplied from an operator for execution of the program 4. Then the program 1 reads the working environment flag of the program 4 out of an entry table having the same function contents as said input ones. The program 1 secures an OR of each bit of the working environment flag for each bit of the read-out working environment condition flag of the program 4. Then the relevant program 4 is carried out when each OR result is not equal to '0'. Thus the program 1 can decide the propriety for execution of the programs 4 just by carrying out a process turned into a pattern where the standardized bit groups, that is, the working environment flag and the working environment condition flag are compared with each other.

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　昭64－10328

⑤1Int.Cl.⁴	識別記号	庁内整理番号	⑬公開　昭和64年(1989)1月13日
G 06 F 9/06	310	A－7361－5B	

審査請求　未請求　発明の数 1　（全6頁）

⑤4発明の名称　プログラム実行管理方式

⑳特　願　昭62－165368
㉒出　願　昭62(1987)7月3日

⑦2発明者	田丸　久	神奈川県秦野市堀山下1番地　日立コンピュータエンジニアリング株式会社内
⑦2発明者	武末　八太力	神奈川県秦野市堀山下1番地　株式会社日立製作所神奈川工場内
⑦1出願人	株式会社日立製作所	東京都千代田区神田駿河台4丁目6番地
⑦1出願人	日立コンピュータエンジニアリング株式会社	神奈川県秦野市堀山下1番地
⑦1出願人	日立電子サービス株式会社	神奈川県横浜市戸塚区品濃町504番2号
⑦4代理人	弁理士　小川　勝男	外1名

最終頁に続く

明　細　書

1　発明の名称

プログラム実行管理方式

2　特許請求の範囲

1　機能分割された複数の下位プログラムと、該下位プログラム群を制御する上位プログラムとから成るプログラムにおいて、上位プログラムと下位プログラムとをリンクさせる第1のステップと、プログラムを実行中のコンピュータシステムの動作環境を示す動作環境と各下位プログラムの動作条件を示す動作環境条件と、を比較することにより下位プログラムの実行を許可するか否かを判定する第2のステップを設けたことを特徴とするプログラム実行管理方式。

3　発明の詳細な説明

〔産業上の利用分野〕

本発明はコンピュータシステムに係り、特に複数の処理機能を有するプログラムが複数の環境で動作するプログラム実行管理方式に関する。

〔従来の技術〕

近年、コンピュータシステムの高度化に伴い、プログラムの規模、機能が、増大、多機能化する傾向にある。しかしながら、単一のプログラムをそのまま大規模化し、多機能化すると、デバッグ作業、エンハンス作業の増加、ひいてはプログラムの品質低下を引き起こすことが知られている。この為、プログラム設計においては、機能分割／階層構造等の技法、すなわち、単一機能を持つ下位プログラムを、上位プログラムが選択／実行する技法を採ることが一般的となっている。一方、ハードウエアの面に於いても、新技術、新機能、新アーキテクチャの開発／実装がくり返されており、この結果プログラムには、様々な環境で動作する汎用性が求められている。

なお、この種の公知例として例えば特開昭61－123936号公報が挙げられる。

〔発明が解決しようとする問題点〕

機能分割／階層構造的技法を採るプログラムが様々な環境下で動作できる汎用性を持つ為には、以下に示す様な問題がある。

すなわち、上位プログラムは不特定の環境下で動作可能であることが要求される一方、機能分割され特定の機能しか持たない下位プログラムの中には、必然的に、特定の環境下でしか動作しないものが発生するようになる。この為、上位プログラムから下位プログラムへ制御を移す際、実行中の動作環境で下位プログラムが動作可能か否かを判定する処理が必要となる。

この判定処理を下位プログラムが自ら行うのは、前述した機能分割の考え方に反する。従って、上位プログラムがこの判定処理を行うのであるが、動作環境が拡大するにつれて判定処理が複雑になるうえ、下位プログラムの追加、削除、変更の度に上位プログラムの処理を修正するのでは、プログラムのエンハンス作業が増大してしまう。

本発明の目的は、以上の如き問題を除去するものであり、様々な動作環境下で、それぞれの下位プログラムの動作環境への適合を上位プログラムが判定し、かつ、この判定方法が下位プログラムの追加、削除、変更に影響されない様な手段をプログラムに与えて、汎用性を持つ階層化プログラムの保守性を高めるプログラム実行管理方式を提供することにある。

〔問題点を解決するための手段〕

上記目的は、プログラムを実行中のコンピュータシステムの動作環境を示す動作環境フラグを設け、さらに、各下位プログラムのエントリテーブル内に、それぞれの下位プログラムに許される動作環境を示す動作環境条件フラグを設け、上位プログラムが下位プログラムを呼び前に動作環境フラグと動作環境条件フラグとの論理和を取り、下位プログラムの動作環境への適合性を判定することにより、達成される。

〔作用〕

プログラム設計者は、あらかじめプログラムの動作環境を分類／規格化したうえで、動作環境フラグと動作環境条件フラグの各ビットにこれらの意味を持たせる。さらに、動作環境フラグは、プログラム動作中の環境を示すものであるから、上位プログラムが下位プログラムの動作環境への適合性を判定する処理とは無関係に設定できる。一方、動作環境条件フラグは、下位プログラムが動作出来る環境を示すものであるから、下位プログラム作成時に設定できる。

これにより上位プログラムは、動作環境フラグと動作環境条件フラグがそれぞれ持つ規格化されたビット群同士を比較すると云うパターン化された処理を行うだけで、不特定の動作環境下において、様々な動作環境条件を持つ下位プログラムの適合性を判定でき、下位プログラムの追加、削除、変更の際、上位プログラムの判定処理を修正する必要はない。

〔実施例〕

以下、本発明の一実施例を図面を用いて詳細に説明する。

第1図は、プログラム全体の構成例を示す図である。

上位プログラム1はオペレータの指示に従って下位プログラム4の中から1つを選択し、動作環境への適合性を判定した後、実行する。その際に、上位プログラム1は動作環境フラグ2とエントリテーブル群3を参照する。動作環境フラグ2は、3バイトのビットデータであり、本プログラム実行中の動作環境を示す。

第2図はエントリテーブル群3の構成例を示す図である。

エントリテーブル群3は下位プログラム4に1対1に対応するエントリテーブル5の集合である。エンドサイン6は、必要なエントリテーブルを検索する処理に終了条件を与えるものである。

第3図はエントリテーブル5の構成例を示す図である。

ネクストポインタ51は4バイトのアドレスデータであり、次のエントリテーブルの先頭アドレスを示す。エントリアドレス52は4バイトのアドレスデータであり、当該エントリテーブル5に対応する下位プログラム4の入口アドレスを示す。動作環境条件フラグ53は3バイトのビットデータであり、当該エントリテーブル5に対応する下位プログラム4が動作出来る動作環境を示す。レ

ングス54は1バイトの16進データであり、後に続く機能内容55のバイト長を示す。機能内容55は不定長の文字データであり、オペレータがコンソールから入力するフォーマットで、当該エントリテーブル5に対応する下位プログラム4の機能を示す。

第4図は動作環境フラグ2と動作環境条件フラグ53の構成例と各ビットの意味を示す図である。ビット0からビット7は本プログラムのユーザの種類を示すビット群である。動作環境フラグ2において、これらのビットの内、本プログラム実行中のユーザに対応する1ビットのみが1であり他のビットは全て0である。動作環境条件フラグ53において、これらのビットの内、当該エントリテーブル5に対応する下位プログラム4の実行を許されるユーザに対応するビットの全てが1であり他のビットは0である。なお、ビット5からビット7は将来の拡張用に用意したビットである。

ビット8からビット15は本プログラムを実行するCPUモードを示すビット群であり本実施例に示すプログラムは6種類のCPUモードで動作できる。動作環境フラグ2においては、これらのビットの内、本プログラム実行中のCPUモードに対応する1ビットのみが1であり他のビットは全て0である。動作環境条件フラグ53においては、当該エントリテーブル5に対応する下位プログラム4が動作出来るCPUモードに対応するビットの全てが1であり他のビットは0である。なお、ビット14とビット15は将来の拡張用に用意したビットである。

以下、同様にビット16，ビット17はCPUがマルチプロセッサかシングルプロセッサかを示すビット群であり、ビット18，ビット19はシステムがイニシャライズ処理中か否かを示し、ビット20，ビット21はシステムがエラー処理中か否かを示す。なお、ビット22，ビット23は将来の拡張用に用意したビットである。

第5図は、上位プログラム1の動作例を示すフローチャートである。以下、第5図を用いて本実施例の動作を詳細に説明する。

入力処理（61）では、オペレータがコンソールから入力した処理要求を受けとる。この処理要求は不定長の文字データとして上位プログラム1に渡されるものである。

検索処理（62）では、上記処理内容と同じ機能内容55を持つエントリテーブル5をエントリテーブル群3の中から探す。

判定1（65）は、検索処理の結果を判定する。即ち、オペレータが入力した処理内容と同じ機能内容55を持つエントリテーブル5がエントリテーブル群3の中に無ければ、エラー処理1（631）において、オペレータ入力エラーを意味するメッセージをコンソールに出力し、処理を終了する。一方、エントリテーブル見つかった場合は、そのエントリテーブルに関して、以下に続く判定2（64）から判定6（68）を行なう。この一連の処理を判定処理（60）とする。

判定2（64）では、動作環境フラグ2のビット0からビット7までと、検索処理（62）で見つけたエントリテーブル5の中の動作環境条件フラグ53のビット0からビット7までとの論理和を取り、結果が0でなければ判定3（65）へ進む。もし、本プログラム実行中のユーザが、検索処理（62）で見つけたエントリテーブル5に対応する下位プログラム4の実行を許されていなければ、論理和の結果は0となり、エラー処理2（641）において、ユーザエラーを意味するメッセージをコンソールに出力し、処理を終了する。

判定3（65）では、動作環境フラグ2と動作環境条件フラグ53のビット8からビット15同士の論理和を取り、結果が0でなければ判定4（66）へ進む。もし、本プログラム実行中のCPUモードが、当該エントリテーブル5に対応する下位プログラム4を実行できないCPUモードであれば、上記論理和の結果は0となり、エラー処理3（651）において、CPUモードエラーを意味するメッセージをコンソールに出力し、処理を終了する。

以下、同様に、判定4（66）、判定5（67）、及び判定6（68）はそれぞれ、当該エントリテー

ブル5に対応する下位プログラム4が、本プログラム実行中の動作環境（マルチプロセッサシステムか否か、イニシャライズ中か否か、及びエラー処理中か否か）において、実行を許可されているか否かの判定を、動作環境フラグ2と動作環境条件フラグ53の意味付けられたビット群同士（ビット16からビット17，ビット18からビット19，及びビット20からビット21）の論理和を取ることにより行ない、許可されていれば、それぞれの次の処理へ進み、許可されていなければそれぞれのエラー処理（661，671，681）を実行して終了する。

実行（69）は、当該エントリテーブル5のエントリアドレス52を用いて、対応する下位プログラム4を呼び出し実行した後、処理を終了する。

以上の動作説明で明らかなように、本実施例によれば、オペレータがコンソールから入力した処理要求を処理する機能を有する下位プログラム4が本プログラム実行中の動作環境下で動作可能か否かを判定する一連の判定処理（60）は、動作環境フラグ2と動作環境条件フラグ53の意味付けられたビット群同士の論理和を取る処理のみで成される。従って、上位プログラム1は、下位プログラム4の追加、削除、変更には影響されず、プログラムの保守性が向上できる。

一方、下位プログラムの追加、削除、或いは変更に於いては、エントリテーブル5の追加、削除、或いは動作環境条件フラグ53の変更を行なうのみで良く、下位プログラム4内の処理で動作環境を考慮する必要がない為、機能分割／階層構造的技法が実現可能となる。

更に、判定処理（60）がパターン化できた事により、オペレータの処理要求を本プログラム実行中の動作環境で実行できない場合のエラー処理2（641）からエラー処理6（681）をサブプログラムとして体系化できる。

本実施例は、処理要求をオペレータが入力するものであったが、他プログラム或いは他システムからの入力を処理するものであっても良いことは明らかである。

また、処理要求は不良長の文字データとしたが、コード化されたビットデータであっても良く、更に、第3図の機能内容55も同じくコード化されたビットデータで良いから固定長となる為、レングス54は不要である。

また、本実施例では、動作環境を分類／規格化したビット群で表わす為、判定1（65）及び判定処理（60）は全てビット群同士の論理和を取ることとしたが、通常の比較演算等で行なっても良い。この場合、動作環境フラグ2と動作環境条件フラグ53のビット構成は同じでなくとも良い。

さらに本実施例では、エラー処理1（631）からエラー処理6（681）においてエラーメッセージをコンソールに出力するが、エラー原因をコード化し報告するだけでも良い。更に、エラー処理2（641）においては、ユーザエラーを意味するメッセージをコンソールに出力したが、エラー処理1（631）と同様にオペレータエラーとして扱うことにより、特定の下位プログラム及びその機能を特定のユーザから隠すことが容易に可能となる。

なお、本実施例においては、本プログラムの動作環境は5種類であるため、5種類のビット群を5回の論理和により判定したが、動作環境に新たな種類が追加されるならば、それに合わせて動作環境フラグ2と動作環境条件フラグ53の未使用のビットに新たな意味付けをし、判定処理とエラー処理を追加すれば良い。また、本実施例においては、動作環境を表わすフラグを最大24ビットとしたが、必要に応じてこのビット数を増す事も容易である。

なお、ビット単位の替わりにバイト単位で扱かってもよい。

〔発明の効果〕

本発明によれば、機能分割／階層構造的技法を用いて設計されたプログラムの上位プログラムは、動作環境フラグと動作環境条件フラグがそれぞれ持つ規格化されたビット群同士を比較すると云うパターン化された処理を行なうだけで、不特定の動作環境下で、様々な動作環境条件を持つ下位プ

ログラムの適合性を判定でき、下位プログラムの追加、削除、変更の際、上位プログラムの上記判定処理を修正する必要がなく、プログラムの機能の追加、削除、変更に対して保守性を高めることができるという効果がある。

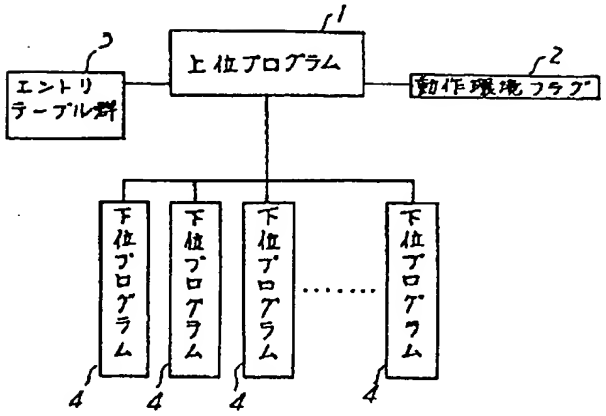
4. 図面の簡単な説明

第1図は本発明の一実施例を示すプログラム構成を示す図、第2図は第1図のエントリテーブル群の構成を示す図、第3図は第2図のエントリテーブルの構成を示す図、第4図は動作環境フラグ動作環境条件フラグを説明するための図、第5図は本発明の一実施例の動作例を示すフローチャートである。

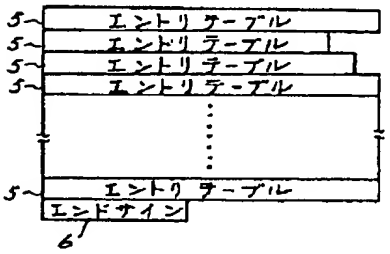
1…上位プログラム、2…動作環境フラグ、3…エントリテーブル群、4…下位プログラム、5…エントリテーブル、51…ネクストポインタ、52…エントリアドレス、53…動作環境条件フラグ、54…レングス、55…機能内容。

代理人弁理士　小　川　勝　男

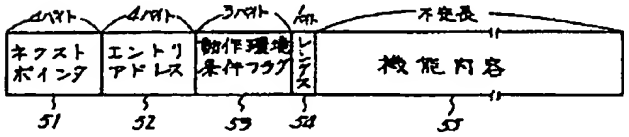
第1図



第2図



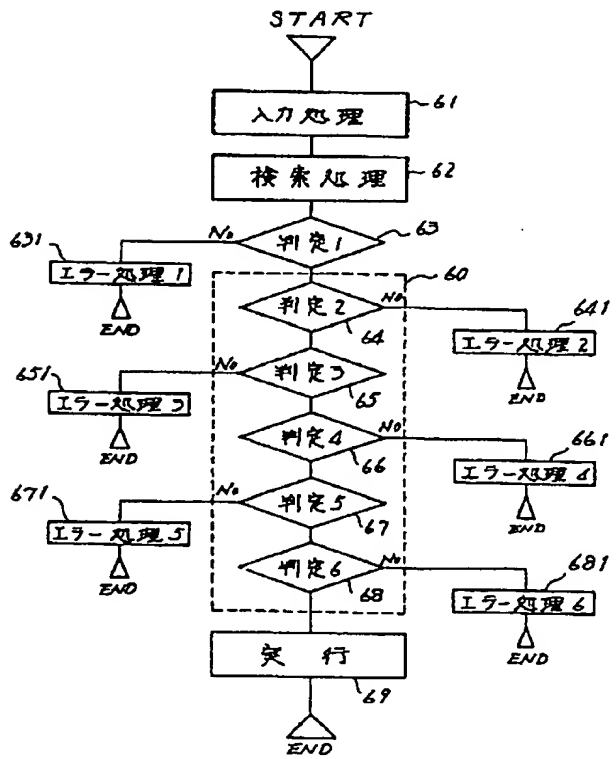
第3図



第4図

バイト	ビット	ビットの意味
0	0	プログラムデバグユーザ
	1	ハードウェアデバグユーザ
	3	フィールドユーザ
	4	ライセンスユーザ
	5	未使用
	6	未使用
	7	未使用
1	8	CPUモード1
	9	CPUモード2
	10	CPUモード3
	11	CPUモード4
	12	CPUモード5
	13	CPUモード6
	14	未使用
	15	未使用
2	16	シングルプロセサシステム
	17	マルチプロセサシステム
	18	イニシャライズ実行中
	19	イニシャライズ完了
	20	ノーマル動作中
	21	エラー処理中
	22	未使用
	23	未使用

第5図



第1頁の続き

⑫発 明 者　小 黒 沢　　利 幸　神奈川県横浜市戸塚区品濃町504番2号　日立電子サービス株式会社内